A Unified Sequential Equivalence Checking Approach to Verify High-Level Functionality and Protocol Specification Implementations in RTL Designs

Carlos Ivan Castro Marquez
Marius Strum
Wang Jiang Chau

School of Engineering, University of São Paulo
Brazil
Agenda

• A glance at verification
• Objectives
• Issues from RTL vs. HL(P) checking
• Formal SEC with HL(P) models
• Application example
• Tool Implementation and Results
• Conclusions
Introduction

• Digital system design consists of several refinement steps

• Complexity increases gradually, making verification essential at every step

• Top-down approaches use golden models as behavioral reference
Introduction

• Equivalence checking:

✓ To verify if two models correspond to the same behavior

✗ Traditionally, it has been applied to descriptions at the same level of abstraction
Introduction

HL vs. RTL Equivalence checking by HL model refinement
  • Koelbl et al (2009)
  • Hao et al (2009)

HL vs. RTL Equivalence checking by DUV abstraction
  • Bombieri et al (2007)
  • Lee et al (2011)
  • Castro et al (2013)
Introduction

The previous works approach to the problem of SEC between different levels of abstraction, however, RTL-specific aspects are left behind.

Elements such as interface protocol, scheduling, and timing must also be verified in order to establish design correctness.
Introduction

The behavior of RTL designs can be divided in:
• computation: treatment of data and control described in the HL reference model.
• communication: interface signals and protocol described only in the DUV

- Communication protocol (signal level)
- HL description (computation/processing)

inclusion / adaptation

refinement

RTL description
Introduction

The behavior of RTL designs can be divided in:
• computation: treatment of data and control described in the HL reference model.
• communication: interface signals and protocol described only in the DUV
Introduction

(a) HL

void gcd()
{
    int x=in1, y=in2;
    while (y!=0)
    {
        while (x>=y) {
            x=x-y;
            aux=x;
            x=y; y=aux;
        }
        out=x;
    }
}

SC_MODULE (gcd) {

    void read()
    {
        if (~reset & load)
            x=in1;
            y=in2;
    }

    void write()
    {
        if (reset)
            swap=0;
        else if (load)
            x=in1; y=in2;
        else if (swap) x=y; y=aux;
        else x=aux;
    }

    void calculate()
    {
        if (y!=0) {
            out=0;
            if (x>=y)
                swap=0;
            else
                swap=1;
        }
        else
            out=x;
    }
}

(b) Protocol (P)

(c) RTL
Objectives

• Formalism for RTL verification by DUV simplification and comparison with:
  – a HL model
  – a model of the protocol specification

• Proof to guarantee absence of false negatives

• Algorithm to:
  – decompose the FSMD representation of the descriptions in
    sequences of states
  – compare the sequences
Issues from RTL vs. HL(P) checking

• Traditional equivalence checking:

  – Given $M_1$ and $M_2$, such that $I_1 = I_2$ and $O_1 = O_2$, two states $s_1 \in S_1$ and $s_2 \in S_2$ are equivalent if, starting from these states, any finite sequence of inputs results in the same sequence of outputs when applied to either $M_1$ or $M_2$.

  – $M_1$ and $M_2$ are equivalent if $I_1 = I_2$, $O_1 = O_2$, and their initial states, $s_{01}$ and $s_{02}$, are equivalent.
Issues from RTL vs. HL(P) checking

• Two issues:
  – Inequality of signals and variables
  – Unpairable states
Issues from RTL vs. HL(P) checking

1. $M_{\text{RTL}}$ may contain a greater number of variables, which means that its alphabet is different from that of $M_{\text{HL}}$ and $M_{\text{P}}$.

\[ V = I \cup O \]

$V_{\text{HL}}$, $V_{\text{RTL}}$, $V \cap$, and $V'_{\text{RTL}}$ (assume $V_{\text{HL(P)}} = V \cap$)
Issues from RTL vs. HL(P) checking

2. Unpairable states

States may be found at either $M_{HL}$ or $M_P$ that have no equivalent state in $M_{RTL}$, and vice versa.
Formal SEC with HL(P) Models

Solutions to the previous problems:

1. Modeling of descriptions as FSMD

2. Checking based on comparison of state sequences

3. Removal of $V'_{\text{RTL}}$ from $V_{\text{RTL}}$

4. Comparison of the resulting sets of sequences
Formal SEC with HL(P) Models

Rules to generate sequences for a FSMD

(i) Control states generate a new sequence for every decision branch (e.g. $k_0$ and $k_1$), unless overridden by rule (iii)

(ii) One sequence may not include more than one state instance, unless it is the last sequence state, or unless overridden by rule (iv)
Formal SEC with HL(P) Models

Rules to generate sequences for a FSMD

(iii) Rule (i) is overridden if there is a variable whose actual value activates one of the decision branches. In such case, no new sequences are created, and the current sequence continues its extension through the activated branch.

(iv) Rule (ii) is overridden if after overriding rule (i), there is a path from the recurring state to the control state.
Formal SEC with HL(P) Models

Any machine $M$ has a set $K$ of state sequences: \{${k_0, k_1, ..., k_n}$\}

The idea is to discover if every $k_{\text{HL}} \in K_{\text{HL}}$ has a functional equivalent in $K_{\text{RTL}}$

Given two sets $K_{\text{HL}}$ and $K_{\text{RTL}}$, a direct comparison is not possible because of the three problems aforementioned
Formal SEC with HL(P) Models

Removal of $V'_{\text{RTL}}$ from $V_{\text{RTL}}$

Reduction function, $R(k_{\text{RTL}}) : V_{\text{RTL}} \rightarrow V \cap ; k^R_{\text{RTL}} = R(k_{\text{RTL}})$

- $V'_{\text{RTL}}$ (wrt $k^R_{\text{RTL}}$) = $\emptyset$

- Three operations:
  - Constant propagation
  - Removal
  - Grouping
Formal SEC with HL(P) Models

Removal of $V'_{RTL}$ ($d,e,f,u,v,w$) from $V_{RTL}$

<table>
<thead>
<tr>
<th>$k_{RTL}$</th>
<th>Transition Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$d=1 / u=1$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$a=u / v=0$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$e=0 / x=u$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$f=0 / b=a, w=1$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$b=u / y=0$</td>
</tr>
<tr>
<td>$s_5$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k^1_{RTL}$</th>
<th>Transition Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$d=1 / u=1$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$a=1 / v=0$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$e=0 / x=1$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$f=0 / b=1, w=1$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$b=1 / y=0$</td>
</tr>
<tr>
<td>$s_5$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k^2_{RTL}$</th>
<th>Transition Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_0$</td>
<td>$\emptyset / \emptyset$</td>
</tr>
<tr>
<td>$s_1$</td>
<td>$a=1 / \emptyset$</td>
</tr>
<tr>
<td>$s_2$</td>
<td>$\emptyset / x=1$</td>
</tr>
<tr>
<td>$s_3$</td>
<td>$\emptyset / b=1$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$b=1 / y=0$</td>
</tr>
<tr>
<td>$s_5$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k^R_{RTL}$</th>
<th>Transition Diagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_{0123}$</td>
<td>$a=1 / x=1,$</td>
</tr>
<tr>
<td>$s_4$</td>
<td>$b=1$</td>
</tr>
<tr>
<td>$s_5$</td>
<td>$b=1 / y=0$</td>
</tr>
</tbody>
</table>
Formal SEC with HL(P) Models

Removal of $V'_{\text{RTL}}$ from $V_{\text{RTL}}$

**Postulate 1.** Two sequences of states, $k_1$ and $k_2$, have the same behavior with respect to a set of variables, $V$, if the following conditions are satisfied:

- $k_1$ and $k_2$ have equal sets of input variables, with the same ordering

- $k_1$ and $k_2$ have equal sets of output variables, with the same ordering

- If $i$ is an input, and $o$ an output, with variables $v_i \in I$ and $v_o \in O$, respectively, then $i$ triggers the same output assignment, $o$, at both $k_1$ and $k_2$
Formal SEC with HL(P) Models

Removal of \( V'_{\text{RTL}} \) from \( V_{\text{RTL}} \)

**Theorem 1.** The operations from function \( R \) do not alter the behavior between \( k_{\text{RTL}} \) and \( k^1_{\text{RTL}} \), \( k^1_{\text{RTL}} \) and \( k^2_{\text{RTL}} \), and \( k^2_{\text{RTL}} \) and \( k^R_{\text{RTL}} \) with respect to \( V \cap \).

**Corollary 1.** Any sequence \( k^R_{\text{RTL}} \) presents the same behavior of a sequence \( k_{\text{RTL}} \), with respect to the set of variables \( V \cap \), as long as \( k^R_{\text{RTL}} \) is obtained by applying the reduction function, \( R \), on \( k_{\text{RTL}} \), i.e., \( R(k_{\text{RTL}}) = k^R_{\text{RTL}} \).

Is corollary 1 enough to apply SEC between HL and RTL descriptions?
Formal SEC with HL Models

Removal of $V'_{\text{RTL}}$ from $V_{\text{RTL}}$

Ideally, there should be warranty of absent false negative verification results.

In order to avoid the complexity involved in such determinism when any two $M_{\text{RTL}}$ and $M_{\text{HL}}$ are considered, the present framework appeals to a “stronger” verification strategy, achieved by restricting the whole set of all possible $k_{\text{RTL}}$.

For that, we start by defining such simplified set of sequences as Extended Sequences.
Formal SEC with HL Models

$k_{HL}$ is extended

$k_{RTL1}$ is extended

$k_{RTL2}$ is NOT extended
The GCD Example

\( M_{\text{RTL}} \)

- \( b_0 \):
  - \( \text{rst} = 1 \):
    - state = 0
    - out = 0
  - \( \text{rstate} = 1 \):
    - state = 0
    - \( \text{out} = x \)
    - state = 0

- \( b_1 \):
  - \( \text{go} = 1 \):
    - x = \text{in1}
    - y = \text{in2}
    - state = 1
  - \( \text{go} = 0 \):
    - state = 0
    - \( \emptyset \)

- \( b_2 \):
  - \( \text{state} = 1 \):
    - \( \text{temp}_x = x - y \)
    - state = 2
  - \( x >= y \):
    - x = \text{temp}_x
    - state = 1

- \( b_3 \):
  - \( x = y \):
    - out = x
    - state = 0

- \( b_4 \):
  - \( \text{state} = 2 \):
    - \( \emptyset \)
The GCD Example

Sequences obtained for $M_{\text{RTL}}$

$k_{\text{RTL0}}$: $b_0 \rightarrow b_0 \rightarrow b_4 \rightarrow b_1 \rightarrow b_0$

$k_{\text{RTL1}}$: $b_0 \rightarrow b_0 \rightarrow b_4 \rightarrow b_1 \rightarrow b_0 \rightarrow b_4 \rightarrow b_0 \rightarrow b_4 \rightarrow b_3 \rightarrow b_0$

$k_{\text{RTL2}}$: $b_0 \rightarrow b_0 \rightarrow b_4 \rightarrow b_1 \rightarrow b_0 \rightarrow b_4 \rightarrow b_0 \rightarrow b_4$

$k_{\text{RTL3}}$: $b_0 \rightarrow b_0 \rightarrow b_4 \rightarrow b_1 \rightarrow b_0 \rightarrow b_4 \rightarrow b_0 \rightarrow b_4 \rightarrow b_3 \rightarrow b_0$
The GCD Example

Protocol specification

Sequences obtained for $M_p$

<table>
<thead>
<tr>
<th>$\mathbf{i}_{E0}$</th>
<th>$\mathbf{o}_{E0}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. rst==1</td>
<td>out = 0</td>
</tr>
<tr>
<td>2. rst==0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>3. go==0</td>
<td>$\emptyset$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\mathbf{i}_{E1}$</th>
<th>$\mathbf{o}_{E1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. rst==1</td>
<td>out = 0</td>
</tr>
<tr>
<td>2. rst==0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>3. go==1</td>
<td>$x=$in1, $y=$in2</td>
</tr>
<tr>
<td>4. rst==0</td>
<td>out = $x$</td>
</tr>
</tbody>
</table>

$M_p$

$p_0$

$rst=0 / out=x$

$p_1$

$go=0 / \emptyset$

$p_2$

$go=1 / x=$in1, $y=$in2

$p_3$

$rst=0 / \emptyset$

$rst=1 / out=0$
The GCD Example

Reducing $k_{RTL1}$ an equivalent sequence to $k_{E1}$ is obtained

<table>
<thead>
<tr>
<th>$i_{RTL1}$</th>
<th>$o_{RTL1}$</th>
<th>$i_{RTL1}'$</th>
<th>$o_{RTL1}'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. rst == 1</td>
<td>state=0, out=0</td>
<td>1. rst == 1</td>
<td>state=0, out=0</td>
</tr>
<tr>
<td>2. rst == 0</td>
<td>∅</td>
<td>2. rst == 0</td>
<td>∅</td>
</tr>
<tr>
<td>3. state == 0</td>
<td>∅</td>
<td>3. state == 0</td>
<td>∅</td>
</tr>
<tr>
<td>4. go == 1</td>
<td>x=in1, y=in2, state=1</td>
<td>4. go == 1</td>
<td>x=in1, y=in2, state=1</td>
</tr>
<tr>
<td>5. rst == 0</td>
<td>∅</td>
<td>5. rst == 0</td>
<td>∅</td>
</tr>
<tr>
<td>6. state == 1</td>
<td>temp_x=x-y, state=2</td>
<td>6. state == 1</td>
<td>temp_x=x-y, state=2</td>
</tr>
<tr>
<td>7. rst == 0</td>
<td>∅</td>
<td>7. rst == 0</td>
<td>∅</td>
</tr>
<tr>
<td>8. state == 2</td>
<td>∅</td>
<td>8. state == 2</td>
<td>∅</td>
</tr>
<tr>
<td>9. x == y</td>
<td>out=x, state=0</td>
<td>9. x == y</td>
<td>out=x, state=0</td>
</tr>
</tbody>
</table>
The GCD Example

Reducing $k_{RTL1}$ an equivalent sequence to $k_{E1}$ is obtained

<table>
<thead>
<tr>
<th>$k_{RTL1}^2$</th>
<th>$i_{RTL1}$</th>
<th>$o_{RTL1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>rst == 1</td>
<td>out=0</td>
</tr>
<tr>
<td>2.</td>
<td>rst == 0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>3.</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>4.</td>
<td>go == 1</td>
<td>x=in1, y=in2</td>
</tr>
<tr>
<td>5.</td>
<td>rst == 0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>6.</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>7.</td>
<td>rst == 0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>8.</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>9.</td>
<td>$\emptyset$</td>
<td>out=x</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$k_{RTL1}^{R'}$</th>
<th>$i_{RTL1}$</th>
<th>$o_{RTL1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>rst == 1</td>
<td>out=0</td>
</tr>
<tr>
<td>2.</td>
<td>rst == 0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>3.</td>
<td>go == 1</td>
<td>x=in1, y=in2</td>
</tr>
<tr>
<td>4.</td>
<td>rst == 0</td>
<td>out=x</td>
</tr>
</tbody>
</table>

$V'_{RTL} = \{ x, y, state \}$
Implemented tool

VHDL/SystemC
DUV’s RTL code

VHDL/SystemC
HL model’s code

VHDL / SystemC Parser

Python Script

DUV’s AST

HL model’s AST

AST → FSMD

M_{HL}

M_{RTL}

M_p (manual)

Sequence extraction

K_{HL}, K_{RTL}, K_p

Comparison between sets of sequences

Dependence analysis

SMT solver (CVC3)
Results

Table 1. Results of checking between $M_p$ and $M_{RTL}$ for VHDL cores.

<table>
<thead>
<tr>
<th>DUV</th>
<th>States in $M_p$</th>
<th>States in $M_{RTL}$</th>
<th>Sequences in $M_p$</th>
<th>Sequences in $M_{RTL}$</th>
<th>Branches in $M_p$</th>
<th>Branches in $M_{RTL}$</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>23</td>
<td>9.8</td>
</tr>
<tr>
<td>gcd</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>11</td>
<td>14.5</td>
</tr>
<tr>
<td>cordic: p2r</td>
<td>6</td>
<td>28</td>
<td>1</td>
<td>193</td>
<td>2</td>
<td>25</td>
<td>45.9</td>
</tr>
<tr>
<td>cordic: r2p</td>
<td>6</td>
<td>49</td>
<td>1</td>
<td>251</td>
<td>2</td>
<td>37</td>
<td>38.1</td>
</tr>
<tr>
<td>mdct</td>
<td>15</td>
<td>36</td>
<td>2</td>
<td>167</td>
<td>8</td>
<td>76</td>
<td>15.6</td>
</tr>
<tr>
<td>aes</td>
<td>11</td>
<td>265</td>
<td>3</td>
<td>256</td>
<td>5</td>
<td>124</td>
<td>36.2</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>DUV</th>
<th>States in $M_{HL}$</th>
<th>States in $M_{RTL}$</th>
<th>Branches in $M_{HL}$</th>
<th>Branches in $M_{RTL}$</th>
<th>Bugs injected in $M_{RTL}$</th>
<th>Bugs detected in $M_{RTL}$</th>
<th>Number of runs</th>
<th>Mean time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>crc</td>
<td>25</td>
<td>29</td>
<td>49</td>
<td>57</td>
<td>50</td>
<td>45</td>
<td>50</td>
<td>89.3</td>
</tr>
<tr>
<td>fec</td>
<td>43</td>
<td>47</td>
<td>70</td>
<td>78</td>
<td>50</td>
<td>48</td>
<td>50</td>
<td>65.4</td>
</tr>
<tr>
<td>fhec</td>
<td>34</td>
<td>42</td>
<td>65</td>
<td>73</td>
<td>50</td>
<td>45</td>
<td>50</td>
<td>81.0</td>
</tr>
<tr>
<td>white</td>
<td>36</td>
<td>44</td>
<td>44</td>
<td>52</td>
<td>50</td>
<td>50</td>
<td>26</td>
<td>56.9</td>
</tr>
<tr>
<td>cipher</td>
<td>21</td>
<td>29</td>
<td>24</td>
<td>32</td>
<td>50</td>
<td>50</td>
<td>33</td>
<td>25.3</td>
</tr>
</tbody>
</table>
Conclusions

• The formalism proposed is applicable to verify formally RTL designs with respect to both HL behavioral and communication protocol models.

• The conditions to avoid false negatives in the proposed verification framework are presented.

• We’ve developed a tool based on an algorithm for machine states comparison and presented some results.
Thank you!

Carlos Ivan Castro Marquez
Marius Strum
Wang Jiang Chau

Laboratory of Microelectronics
Group of Integrated Electronic Systems and Applied Software ()
School of Engineering, University of São Paulo
Brazil